How words can tell what actions are doing[*]

[Short Paper]

Karl-Heinz Krempels
University of Aachen,
Dept. of Computer Science, Informatik IV
Ahornstr. 55
52074 Aachen, Germany
krempels@cs.rwth-aachen.de

Jens Nimis
Universität Karlsruhe (TH),
Institute for Program Structures
and Data Organization
Am Fasanengarten 5
76131 Karlsruhe, Germany
nimis@ipd.uni-karlsruhe.de

ABSTRACT

This paper describes an approach to use embedded descriptions of agent actions in task ontologies and agent communication messages compliant to FIPA[1] ACL [4, 5]. Ontological terms denoting actions are extended with semantical descriptive decorations in such a way that an agent requested to perform a well defined action is able to extract the formal description of the activity bound to this action from the ontology and to execute it, or to forward it to an inference machine. Further an extension of FIPA SL [7] is presented that allows the integration of this approach in existing agent systems.

Categories and Subject Descriptors

1 [**Agent Communication**]: Speech Acts, Content Languages; 5 [**Infrastructure and Architecture**]: Agent Intertask Communication

Keywords

Agent Communication, Ontology, Content Language, Agents

1. INTRODUCTION

Communication in third generation MAS (multi agent systems) is based more and more on FIPA recommended interaction protocols and speech acts, and existing semantic languages [1], such as KIF, Prolog, FIPA Semantic Language, ebXML etc. Within the content of the messages constituting an interaction protocol, ontologies are used for the specification of domain and task oriented terms and relations. Therefore frame based representations are used for the defined concepts inside of an ontology. The ontology design process is well supported by different development and maintenance tools, like e.g. Protégé, Ontolingua, and OilEd. Yet, a comfortable export of ontologies for direct use in existing agents, agent systems and their development process is not supported. Below we describe an approach to extend task ontology definitions of agent actions by semantical descriptive decorations, and the use of these descriptions in MAS by a rule based inference machine. This means that the agent passes the content of received messages to the inference machine for the action evaluation. The description of agent actions should be supported by the used ontology design tool in such a way that code of the description language can be contributed by the ontology designer as well as by the agent developer. The evaluation of agent action descriptions presumes the ability of the agent's inference machine to understand the used content language and the used descriptive language(s). The outline of the paper is as follows: In Section 2 we describe the state of the art for using ontologies and therein defined actions in agent systems. This is followed by the description of our approach, regarding the ontology extension, the requirements for ontology design tools and an example for embedded action descriptions in FIPA SL [7] in Sections 3. We finish the paper with conclusions and an outlook in Section 4.

2. STATE OF THE ART

For a long time the provided interfaces by ontology design tools were limited to data and representation formats, such as XML and RDF, not ready to use in agent systems or their development process. Developers and researchers dealing with the deployment of ontologies in agent systems and with the implementation of FIPA compliant interaction protocols [6] had to create manually a mapping between the well-defined ontology in the design tool and the needed class definitions of ontology terms in the programming environment. The construction of an ontology according to this process was a very time-consuming and error-prone development task. Figure 1 provides an example of such an ontology class definition file used in *JADE*[2] Construction of large ontologies with hundreds of concepts and terms, seemed to

[*]This work is motivated by experiences made in the German priority research programme 1083 *Intelligent Agents in Real-World Business Applications* where two large multi multi agent systems (MMAS) were developed.
[1]Foundation for Intelligent Physical Agents. http://www.fipa.org/

[2]Java Agent DEvelopment Framework. http://jade.cselt.it/

become an unreachable goal without tool support including automated consistency checks.

```
...
public class MusicShopOntology extends Ontology {
// The name identifying this ontology
public static final String
            ONTOLOGY_NAME = "Music-shop-ontology";
// VOCABULARY
public static final String ITEM = "Item";
public static final String
            ITEM_SERIAL = "serial-number";
public static final String CD = "CD";
public static final String CD_NAME = "name";
public static final String CD_TRACKS = "tracks";
...
// Private constructor
private MusicShopOntology() {
    // This ontology extends the basic ontology
    super(ONTOLOGY_NAME, BasicOntology.getInstance())
    try {
         add(new ConceptSchema(ITEM), Item.class);
         add(new ConceptSchema(CD), CD.class);
    ...
         ConceptSchema
              cs = (ConceptSchema) getSchema(ITEM);
         // The serial-number slot is optional and
         // allowed values are integers.
         cs.add(ITEM_SERIAL, (PrimitiveSchema)
              getSchema(BasicOntology.INTEGER),
              ObjectSchema.OPTIONAL);

         // Structure of the schema for the CD concept
         cs = (ConceptSchema) getSchema(CD);
         cs.addSuperSchema((ConceptSchema)
              getSchema(ITEM));
         cs.add(CD_NAME, (PrimitiveSchema)
              getSchema(BasicOntology.STRING));
         // The tracks slot has cardinality > 1
         cs.add(CD_TRACKS, (ConceptSchema)
              getSchema(TRACK), 1,
         ObjectSchema.UNLIMITED);
    ...
    } catch (OntologyException oe) {
         oe.printStackTrace();
    }
...
```

Figure 1: Part of an ontology class definition in JADE 2.5

Support for automated generation of ontology class definition files, as shown in Figure 1 is provided for the ontology design tool *Protégé*[3] through the *Bean Generator*[4] plugin [8]. Thenceforth, the export of existing large ontologies as Java code became possible and the way to implement them in agents and agent systems was simplified but not solved.

Hence, the new challenge seems to be the implementation of complex queries with respect to the defined ontology, the used content language and the programming language of the used agent systems. In most FIPA compliant agent systems Java is used as programming language and SL [7] as a content language. Ontologies therefore are usually designed and maintained, e.g. with Protégé, and exported to a target language, e.g. as Java code with Protégé's Bean Generator plugin. From these three facts, there results a new challenge for developers, because now they have to create the message content for agent communication further by hand in Java, and they have to keep in mind the generated structure of the defined ontology and the SL grammar rules. The construction of a simple query becomes a very expendable and time-consuming job, due to the need to define the used terms at all SL grammar levels, to instantiate them, and to initialize them with the right sub-terms manually. (To get an idea of this task, think of a bottom-up approach and consider the grammar of SL [7][pages 2-4]: *VariableIdentifier*, *Variable*, *Wff.*, etc.)

For real intelligent agents complex queries are essential and we are allowed to assume that they have to cover more than the trivial case, consisting of two variables and boolean logic. Furthermore the used ontological concepts have to be instantiated and initialized as well as their defined slots and integrated in the described SL term. An example for this is shown in Figure 2 [2][pages 20-21].

```
...
// Prepare the Query-REF message
ACLMessage msg =
   new ACLMessage(ACLMessage.QUERY_REF);
// sellerAID is the AID of the Seller agent
msg.addReceiver(sellerAID)
msg.setLanguage(codec.getName());
msg.setOntology(ontology.getName());
// Prepare the content.
try {
  AbsConcept absCd = ontology.fromObject(cd);
  AbsVariable x = new AbsVariable({\‘O}x{\’O},
     BasicOntology.INTEGER);
  AbsPredicate absCosts =
     new AbsPredicate(MusicShopOntology.COSTS);
  absCosts.set(MusicShopOntology.COSTS_ITEM, absCd);
  absCosts.set(MusicShopOntology.COSTS_PRICE, x);
  AbsIRE absIota = new AbsIRE(SLVocabulary.IOTA);
  absIota.setVariable(x);
  absIota.setProposition(absCosts);
  // Let JADE convert from Abstract descriptor
  // to string
  getContentManager().fillContent(msg, absIota);
  send(msg);
} catch (CodecException ce) {
    ce.printStackTrace();
} catch (OntologyException oe) {
    oe.printStackTrace();
}
...
```

Figure 2: Creation of a QUERY-REF message in JADE 2.6

But this is half the work to be done, because a sent mes-

[3]http://protege.stanford.edu/
[4]http://www.swi.psy.uva.nl/usr/aart/beangenerator/

sage has to be checked syntactically and semantically on the receiving side. Therefore at first the message is passed to a parser for syntax-checking and then the used ontological concepts have to be extracted by hand. If the received message is a query then the used operators have to be mapped to corresponding implementation language operators or the needed logic has to be implemented as well. If the received message is an request for a specific action, then at this place the term specified for an agent action in the used ontology is bound, e.g to a JAVA method that executes this action. This again is a painful job that in the AI field usually is done by inference machines and not by developers. One possibility to simplify this process is to create templates for message contents with the ontology design tool Protégé and the Bean Generator plugin, but this is not a very sophisticated solution because intelligent agents should be able to generate queries and requests at run-time.

After we have described the way in which at present we actually can use ontologies in agent systems and in which we have to program messages for agent communication, we propose a new approach in the following section.

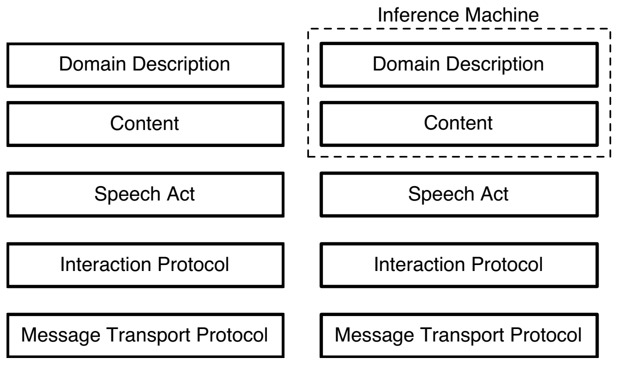
3. ONTOLOGY, MAS AND DESIGN TOOL EXTENSIONS

From the described problems, regarding the use of ontologies in agent systems and the creation of queries, we recognize the following requirements to a new approach:

- Agent actions defined in an ontology should be extended with semantical descriptions consisting of a language identifier and code written in this language.
- The content of received messages should be evaluated within the receiving agent by an inference machine.
- Ontologies should be provided in a language interpretable by the inference machine.

When these requirements are accomplished, the way of developing an interaction among two agents will change from the one shown in Figure 3(a), where all the messages received by an agent are passed from the lowest to the highest level by the agent developer, to the way shown in Figure 3(b). In the new approach represented there, the way of developing an agent interaction will be the following:

- The task ontology used for the domain description is designed, using an ontology tool.
- Semantical action descriptions are added to the defined agent actions by the MAS developers in this ontology design tool.
- The ontology is exported in a language interpretable by the MAS inference machine, while action descriptions have to be embedded therein in their destination programming language representation.
- The ontology is loaded by the agents into their inference machine and herein the agent action descriptions are evaluated.



(a) Ordinary MAS architecture communication stack

(b) Proposed MAS architecture communication stack

Figure 3: Communication stacks in MAS

- Depending on the language specified in an action descriptions name the action descriptions code is forwarded to the respective language processor or it is translated to the language of the agents inference machine [9].

To realize this approach, modifications are necessary regarding ontology definition, MAS architecture and content language specification. Below, possible modifications of task ontologies and of the used design tool Protégé are provided, as well as an adaption of the used content language FIPA SL [7]. The changes necessary in the used MAS framework JADE are negligible, because the content of a received FIPA ACL message can simply be extracted and passed to the agents inference machine.

3.1 Embedded Action Descriptions in Task Ontologies

To integrate support of semantical agent action descriptions into existing ontologies, there are only few changes necessary. At first, a new abstract concept `Description` has to be added to the ontology. Then the slots `name`, `content` and `version` of type `String` have to be added to this concept. Further the meta-concept of all the action concepts, e.g. `action`, has to be extended by a slot with cardinality n of type `description`, instance of the previously defined concept `Description`. Now, when a sub-concept of the extended action concept is instantiated, the new instance can be tagged with semantical descriptions.

3.2 Requirements to Ontology Design Tools

The functionality of the ontology design tool Protégé can be extended by means of plugins. Support for the described approach can be provided through a plugin that enables Protégé to export an ontology to any language interpretable by the MAS inference machine. The best way, probably will be to support a standardized language, such as FIPA SL and to provide language adaptors for different inference machines. This way allows the use of only one plugin and the separated implementation of adapters for every single inference machine. Currently we are working on Protégé plugins for JESS, the Mandarax System[5] and FIPA SL. Additionally language adaptors from FIPA SL to JESS and from FIPA SL to Mandarax are developed. Reasons for this selection are that both systems are implemented in JAVA and support backward reasoning, while their license models are very different. Furthermore, research results in the agent technology area are available for both systems [3].

3.3 Embedded Action Descriptions in FIPA SL

FIPA SL is very similar to the knowledge representation language KIF[6] but there aren't any implementations of inference machines based on it [9]. Unfortunately, the latest changes in FIPA SL are not very elegant, because the chosen names for left hand terms, like e.g. `TermOrIE` (Term or IdentifyingExpression), `Wff` (Well formed formula), do not describe the semantical meaning of the nodes.

```
ActionExpression = "(" "action" Agent TermOrIE ")"
  | "(" "|"  ActionExpression ActionExpression ")"
  | "(" ";"  ActionExpression ActionExpression ")".
```

Figure 4: FIPA SL grammar syntax definition for `ActionExpression`

The current definition of agent action expressions in FIPA SL is shown in Figure 4 and the needed changes to support the presented approach are shown in Figure 5. An implementation of a parser supporting the proposed changes is available and the described language adapters are under development.

```
ActionExpression = "(" "action" Agent
                     ActionDescription* TermOrIE ")"
  | "(" "|"  ActionExpression ActionExpression ")"
  | "(" ";"  ActionExpression ActionExpression ")".
ActionDescription = "(" DescriptionLanguage
                        DescriptionName?
                        DescriptionVersion
                        DescriptionContent ")".
DescriptionLanguage = "(" "description-language"
                        DescriptionSymbol ")".
DescriptionVersion = "(" "description-version"
                        DescriptionSymbol ")".
DescriptionContent = "(" "description-content"
                        DescriptionSymbol ")".
DescriptionSymbol =  String.
```

Figure 5: Proposed modifications in FIPA SL grammar syntax definition for `ActionExpression`

4. CONCLUSIONS AND OUTLOOK

In our paper we outlined an approach to extend ontological terms denoting agent actions with semantical descriptive decorations. This allows for an agent to extract the formal definition of an activity bound to an action it has been requested to perform. It was illustrated how the approach based on the agent internal use of an inference machine can be embedded to the common ontology development tool Protégé and how FIPA SL has to be adapted to support it.

The approach provided is one more step on the way to tool-supported deployment of intelligent agents, and tries to fuel the discussion on the appropriate description and specification format necessary for these tools. The vision is to provide the possibility in one tool, to design the domain and task ontology and to export both as T-boxes in the KB of an agent, to describe a real world scenario with instances of the defined ontologies and to export them in an A-box of the same KB, to create then instances of agents containing an inference machine on the top of the existing KB, and finally, to provide real world scenarios based on distributed MAS.

[5]http://www.mandarax.org/
[6]http://logic.stanford.edu/kif/dpans.html

5. ADDITIONAL AUTHORS

Additional authors:
Lars Braubach (University of Hamburg, email: braubach@informatik.uni-hamburg.de),
Rainer Herrler (University of Würzburg, email: herrler@informatik.uni-wuerzburg.de) and
Alexander Pokahr (University of Hamburg, email: pokahr@informatik.uni-hamburg.de).

6. REFERENCES

[1] L. Botello, S. Willmott, T. Zhang, and J. Dale. Multilingual agents: Ontologies, languages and abstractions. Technical report no. 01/362, Swiss Federal Institute of Technology (EPFL), Lausanne (Switzerland), 2001.

[2] G. Caire. Jade tutorial - application-defined content languages and ontologies. Manual, TILAB, 2002.

[3] J. Dietrich, A. Kozlenkov, M. Schroeder, and W. G. Rule-based agents for the semantic web. *Journal on Electronic Commerce Research Applications*, 2003.

[4] FIPA. Fipa acl message structure specification. Standard, Foundation for Intelligent Physical Agents, http://www.fipa.org/, December 2002.

[5] FIPA. Fipa communicative act library specification. Standard, Foundation for Intelligent Physical Agents, http://www.fipa.org/, December 2002.

[6] FIPA. Fipa iterated contract net interaction protocol specification. Standard, Foundation for Intelligent Physical Agents, http://www.fipa.org/, December 2002.

[7] FIPA. Fipa sl content language specification. Standard, Foundation for Intelligent Physical Agents, http://www.fipa.org/, December 2002.

[8] C. van Aart, R. Pels, G. Caire, and F. Bergentini. Creating and using ontologies in agent communication. *Workshop on Ontologies in Agent Systems, 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2002.

[9] S. Willmott, I. Constantinescu, and M. Calisti. Multilingual agents: Ontologies, languages and abstractions. In *First International Workshop on Ontologies in Agent Systems, Autonomous Agents 2001, Montreal, Canada*, pages 77–84, mai 2001.